

FIG. 1 PRIOR ART

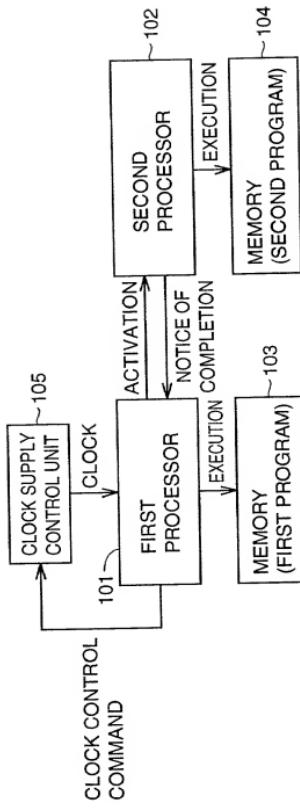


FIG. 2

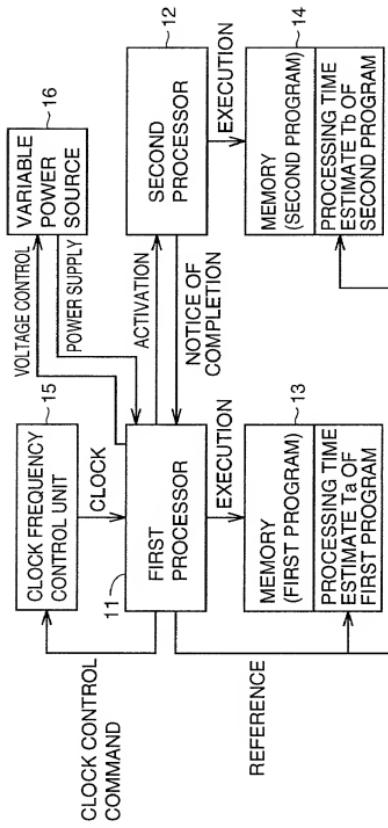


FIG. 3

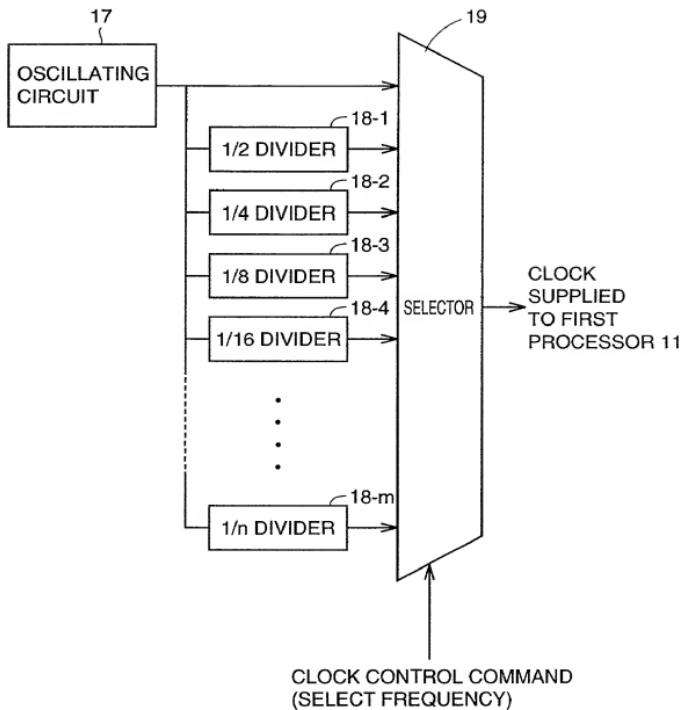


FIG. 4

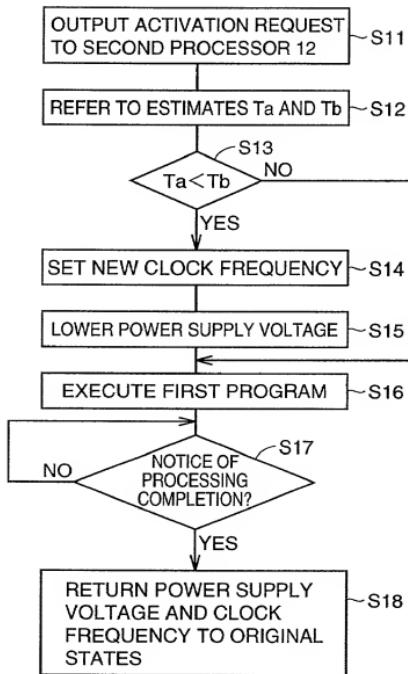


FIG. 5

```
main_procedure()
{
    B_is_done=0;
    invokeB();
    /* ACTIVATE SECOND PROGRAM. PROCESSING TIME ESTIMATE IS Tb */
    set_clock_frequency(Ta/Tb); /* LOWER CLOCK FREQUENCY WITHIN LIMIT DETERMINED BY Ta/Tb */
    set_power(table);
    do_something();
    /* WAIT FOR COMPLETION OF SECOND PROGRAM */
    while (B_is_done)
        /* do nothing */;
}

interrupt_from_B()
{
    restore_power();
    restore_clock_frequency();
    B_is_done=1;
    /* INTERRUPT HANDLING ROUTINE OF FIRST PROGRAM */
    /* RETURN VOLTAGE TO ORIGINAL STATE */
    /* RETURN CLOCK FREQUENCY TO ORIGINAL STATE */
}
```

1047860 8582660

FIG. 6

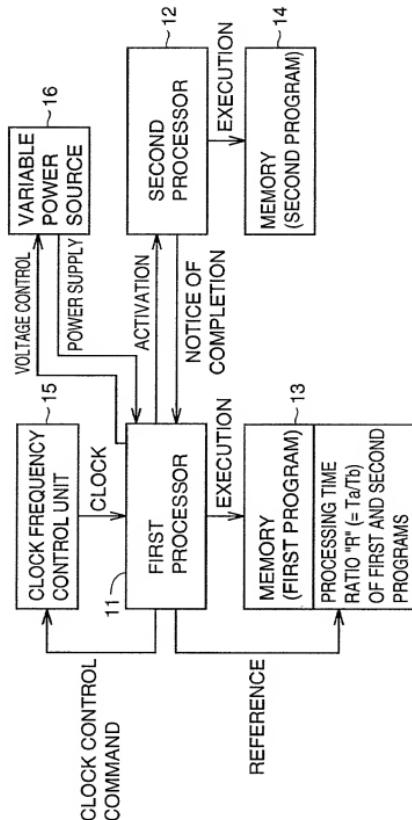


FIG. 7

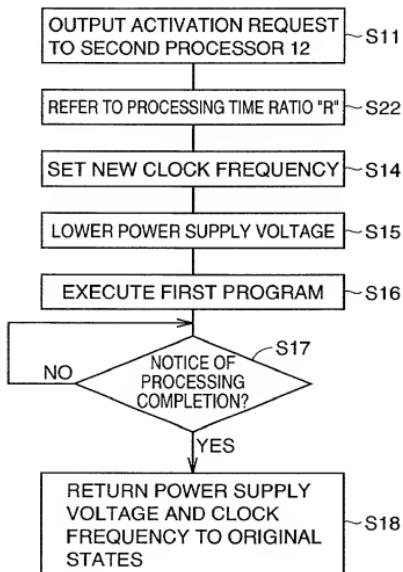


FIG. 8

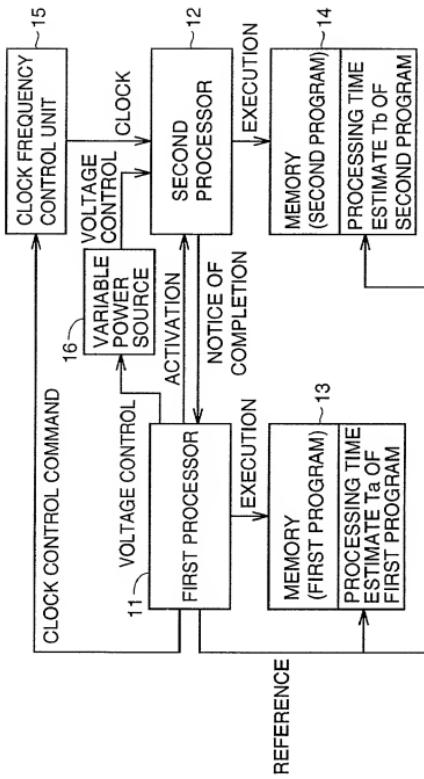


FIG. 9

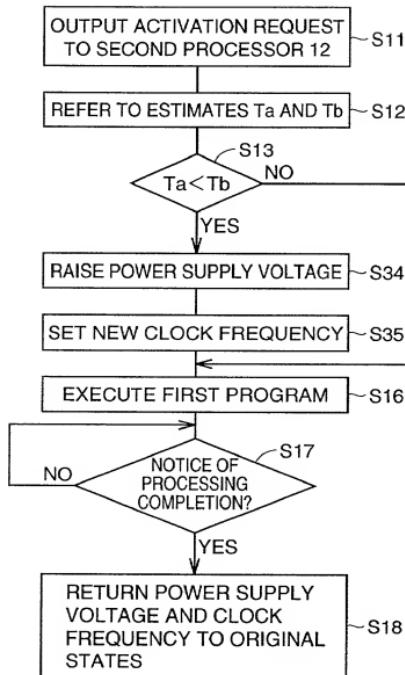


FIG. 10

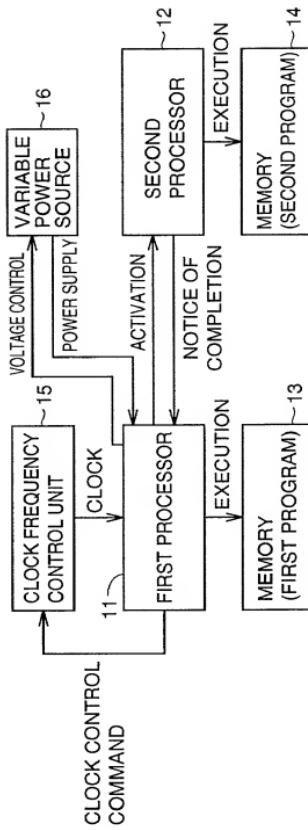
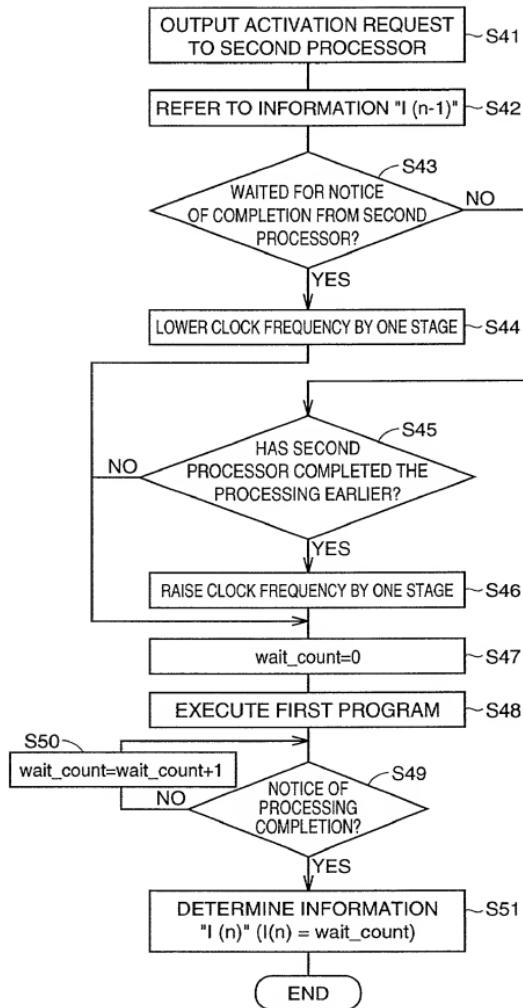


FIG. 11



009923358 - 001414

FIG. 12

```

main_procedure()
{
    B_is_done=0;
    invokeB();
    /* MAIN PROCESSING UNIT OF FIRST PROGRAM */

    /* ACTIVATE SECOND PROGRAM */
    /* IF COMPLETION OF SECOND PROGRAM WAS WAITED IN PREVIOUS PROCESSING, */
    /* LOWER CLOCK FREQUENCY BY ONE STAGE */
    /* IF SECOND PROGRAM WAS COMPLETED EARLIER, */
    /* RAISE CLOCK FREQUENCY BY ONE STAGE */
    /* IF FIRST AND SECOND PROGRAMS WERE COMPLETED APPROXIMATE AT THE SAME TIME, */
    /* DO NOTHING AND KEEP CURRENT CLOCK FREQUENCY */

    if(wait_count>WAIT_LIMIT)
        set_clock_faster();
    else if(wait_count==0)
        set_clock_slower();
    else
        /* do nothing */;

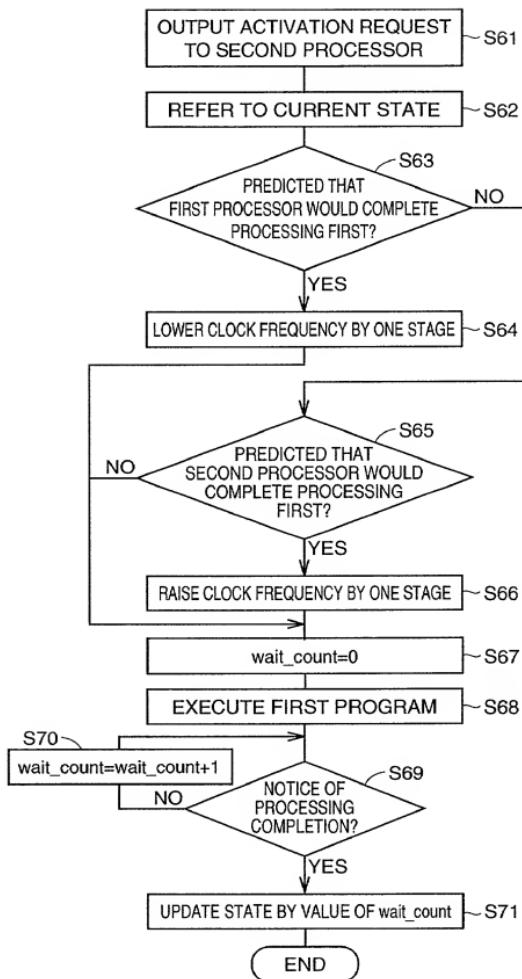
    wait_count=0;
    do_something();
    while(B_is_done==0)
        wait_count++;

}

interrupt_from_B()
{
    B_is_done=1;
}
/* INTERRUPT HANDLING ROUTINE OF FIRST PROGRAM */

```

FIG. 13



2014-09-25 22:59:00

FIG. 14

```
{ main_procedure() /* MAIN PROCESSING UNIT OF FIRST PROGRAM */
{
    B_is_done=0;
    invokeB();
    /* ACTIVATE SECOND PROGRAM */

    PREDICTED VALUE = OBTAIN PREDICTED VALUE FROM CURRENT STATE;
    if (PREDICTED VALUE == "PREDICT THAT PROCESSING OF FIRST PROGRAM WILL BE COMPLETED FIRST")
        set_clock_faster(); /* LOWER CLOCK FREQUENCY BY ONE STAGE */
    else (PREDICTED VALUE == "PREDICT THAT PROCESSING OF SECOND PROGRAM WILL BE COMPLETED FIRST")
        set_clock_slower(); /* RAISE CLOCK FREQUENCY BY ONE STAGE */
    else /* PREDICT THAT FIRST AND SECOND PROGRAMS WILL BE COMPLETED APPROXIMATELY AT THE SAME TIME */
        /* do nothing */;
    wait_count=0;

    do_something(); /* PROCESSING TO BE DONE IN FIRST PROGRAM */

    while (B_is_done==0) /* WAIT FOR COMPLETION OF SECOND PROGRAM */
        wait_count++;
    UPDATE STATE BY VALUE OF wait_count;
}

interrupt_from_B() /* INTERRUPT HANDLING ROUTINE OF FIRST PROGRAM */
{
    B_is_done=1;
}
```

FIG. 15

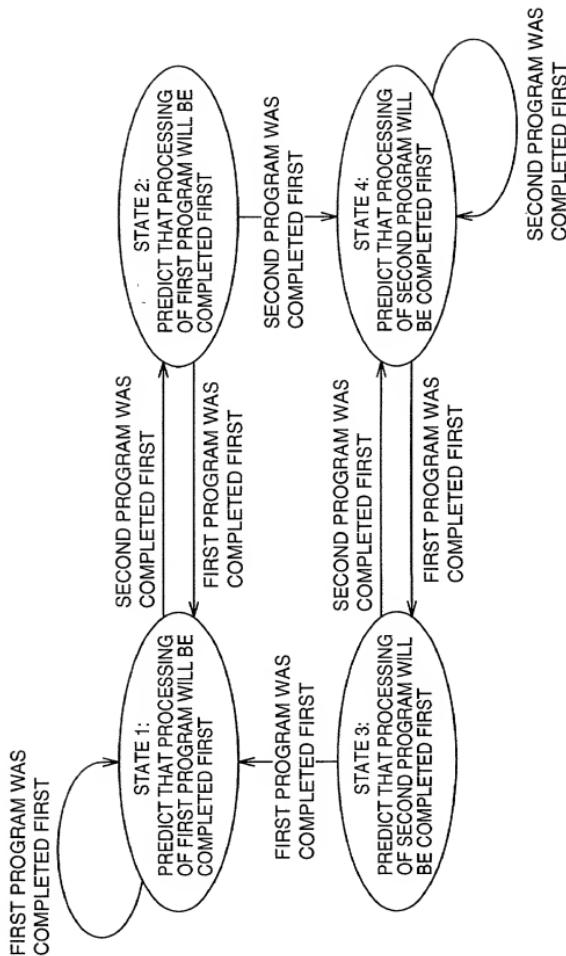


FIG. 16

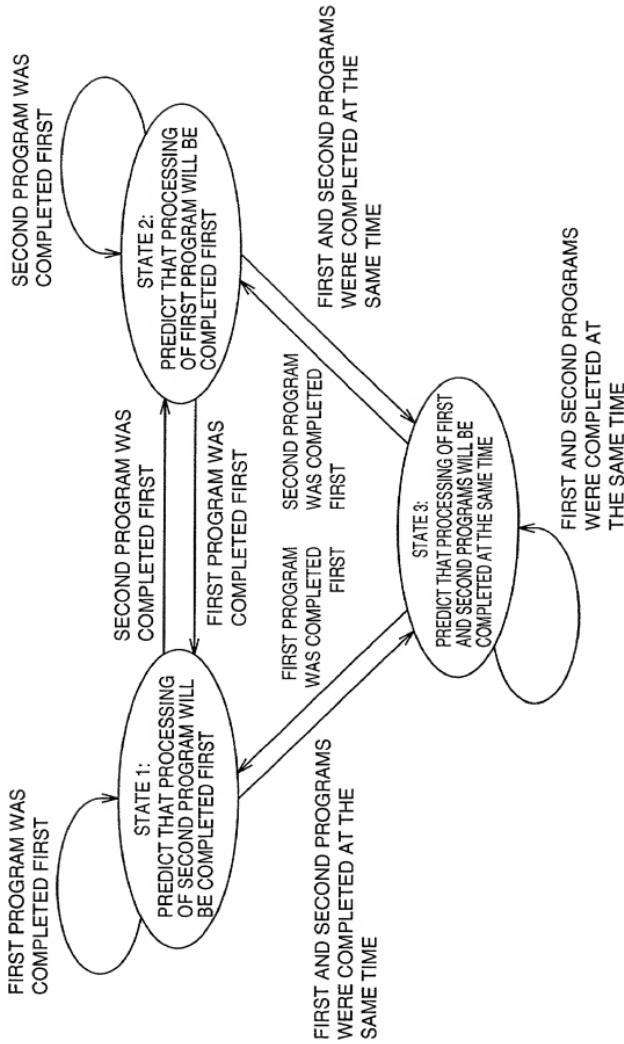


FIG. 17

